Dataset Dynamics Compendium: A Comparative Study

Jürgen Umbrich[1], Boris Villazón-Terrazas[2], and Michael Hausenblas[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway
[2] OEG-DIA, Facultad de Informática, Universidad Politécnica de Madrid, Spain.

Abstract. At the time of writing there exists no consensus about the approaches to detect, propagate and describe changes in resources and datasets of the Linked Open Data Web. This survey gives a comprehensive overview of the current technical solutions and a comparison of such based requirements we derived from use cases the community came up with. We give a detailed overview about the aspects of discovery, granularity level, and description of the changes, as well as the detection algorithms and notification mechanisms. Moreover, we present a high-level dataset dynamics stack that integrates the current technical solutions for dealing with changes in datasets of the Web of Data.

1 On the Importance of Dynamics of Linked Datasets

So far, Linked Data principles and practices are being adopted by an increasing number of data providers, getting as result a global data space on the Web containing billions of RDF triples [8]. However, there are still various research challenges that must be overcome. One particular research challenge is datasets dynamics, and the Linked Data community is realizing the importance of this research area in the last recent months.

Although, there are some efforts to solve this problem in databases, e.g. [28], [18], and [6], among others; they do not cover all the aspects of the Linked Data datasets. This is because linked data can be classified as a self-organising ecosystem, i.e. many units participate in a parallel and distributed manner by creating, publishing and interlinking information. Moreover, enterprises or organisations which are using the paradigm of integrating data by applying the Linked Data principles have to face and deal with the adhering dynamics.

The topic of dataset dynamics covers all kinds of aspects related to changes of and between Linked Data resources and datasets. This research area includes: (1) the design of vocabularies to describe dynamic characteristics and changes of datasets, (2) the auto-discovery of those descriptions, (3) web-scale communication methods for the interaction between consumers and producers for different change granularity levels and (4) algorithms to compute efficiently deltas between two data snapshots.

The players participating in the Linked Data ecosystem are manifold; for instance, the research community, Web 2.0 portals, and Facebook among others. The research community around Linked Data and its efforts to provide tools for data publishers to share their data as Linked Data. The result of these efforts are several software libraries which convert information from arbitrary formats into RDF, for example: D2R Server[3], XLWrap[4], Any23[5], etc.

Also, more and more Web 2.0 portals start to expose their data as Linked Data, for example widely used content management systems like Drupal 7[6] and knowledge

[3] `http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/`
[4] `http://xlwrap.sourceforge.net/`
[5] `http://any23.org/`
[6] `http://drupal.org/node/725382`

management system like SemWiki[7], News portal and multimedia domains, such as the New York Times or BBC, publish their articles and programs according to the principles of Linked Data. Moreover, companies adapt to describe their products using the GoodRelation ontology[8] which leads to better search results for the major search engines like Google or Yahoo. Governments contribute by integrating their data into the LOD cloud. All these players continuously contribute information to the LOD cloud.

The data producers and consumers put a tremendous effort into the interlinking of the valuable information pieces. Furthermore, data publishers continuously try to improve the quality of their data by changing vocabularies, adding new information and relations, or deleting obsolete ones.

Clearly, one can assume and even observe that Linked Data is very dynamic. However, at the time of writing there is no solid solution nor a clear research direction of the big picture of the problem. There are a number of use cases, derived requirements and proposals as we will show in this survey. Nevertheless, we can clearly state that none of the available proposals solve the problem of handling and communicating dataset dynamics in a sufficient way.

The contribution of this work can be summarised as follows: (1) a fundamental overview about the topic of dataset dynamics; (2) the presentation of use cases and requirements agreed by the Linked Data community; (3) a survey and comparison of proposals which are addressing the issues and partial solve them; and (4) an abstract dataset dynamics stack.

Moreover, it is worth to mention that dataset dynamics is important to provide an "efficient" consumption of Linked Data through the discovery, synchronisation, caching, and linkage of the datasets.

The remainder of this paper is organised as follow: Section 2 reviews some existing solutions and Section 3 presents identified use cases. Next, Section 4 discusses requirements derived from the use cases and depicts the high-level architecture of the solution. Then, Section 5 describes the vocabularies and mechanisms that cover the requirements identified. Finally, Section 6 provides some conclusions.

2 Existing Deployed Systems

In this section we review some deployed systems that deal with the dataset dynamics. These systems are the result of preliminary efforts to solve the identified problems of handling and communicating dataset dynamics.

2.1 Sitemap Protocol

Website crawling can be made more efficient and predictable by using the Sitemap Protocol [21], originally developed by Google and now supported by all major search engines, as well as data search engines such as Sindice [24]. It consists of a `sitemap.xml` file that is usually placed in the website root directory and contains a list of all the URLs to be crawled. The Sitemap protocol format consists of XML tags and it defines several elements, being the most importance in our context:

- `url`, entry for each URL, the remaining elements are children of this.
- `changefreq`, which defines how frequently the page is likely to change. This value provides general information to search engines and may not correlate exactly to how often they crawl the page. The sitemap will be fetched with the highest frequency indicated by the URLs contained in it. Because of this, it can save bandwidth if terms with the same change frequency are grouped into separate sitemaps.

[7] `http://km.aifb.kit.edu/ws/semwiki2006/`

[8] `http://www.heppnetz.de/projects/goodrelations/`

– `lastmod`, which represents the date of last modification of the file. This date should be in W3C Datetime[9] format. For example, Sindice uses the `lastmod` element to decide if the given URL has to be re-indexed or not. This is the most important information, as it can reduce the number of requests Sindice will to make to a specific site.

Thanks to the Sitemap protocol, websites that publish RDF datasets are ready for effective discovery and synchronization.

2.2 DBpedia Live

DBpedia is community effort to extract information from Wikipedia and to make this information available on the Web. Nevertheless, a manual effort is necessary to produce a new release and the extracted information is not up-to-date. DBpedia Live [9] is an extension of DBpedia, and is created to tackle the challenging problem of processing tens of thousands of changes per day in order to consume the constant stream of Wikipedia updates. Basically the DBpedia Live provides (1) up-to-date information and (2) a mechanism for allowing the Wikipedia community to maintain the DBpedia ontology collaboratively. The DBpedia Live framework consists of the following main components:

– PageCollections. Abstractions of local or remote resources of Wikipedia articles.
– Destinations. They store extracted RDF triples.
– Extractors. These extractors convert a specific type of wiki markup into triples.
– Parsers. These components help the extractors by identifying datatypes, and converting its corresponding values.
– ExtractionJobs. They consist of a page collection, extractors, and a destination.
– Extraction Manager. This component manages the process of passing Wikipedia articles to the extractors and delivers their output to the destination.

2.3 PubSubHubbub

PubSubHubbub [16] is a decentralized real-time Web protocol that delivers data to subscribers the moment it becomes available. This protocol extends the Atom [17] and RSS [19] protocols for data feeds, basically it turns Atom and RSS feeds into real-time streams.
In a nutshell this protocol has three main participants:

– Publisher, a owner of a topic. It notifies the hub when the topic feed has been updated.
– Hub, a server which implements both sides of this protocol. There are some public hubs, for example: App Engine[10], Superfeedr[11], and RabbitHub[12]
– Subscriber, an entity (program or human being) that wants to be notified of changes on a topic.

Next, we present a very simple example that shows how it works:

1. A blogger or content creator creates a feed and specifies a hub.
2. A consumer subscribes to the blog using the RSS feed in the normal way.
3. New content is created and the source pings the hub saying “i have new content!”
4. The hub in turn “fat pings” the subscribers saying “Hey, the blog has new content, here it is!”

[9] `http://www.w3.org/TR/NOTE-datetime`
[10] `http://pubsubhubbub.appspot.com/`
[11] `http://blog.superfeedr.com/api/http/pubsubhubbub/pubsubhubbub/`
[12] `http://github.com/tonyg/rabbithub/#readme`

2.4 SparqlPuSH

The goal of sparqlPuSH [22] is to enable proactive notification of changes happening in RDF stores, whatever they deal with: new data of a particular type being added, updated statements about a given resource, etc. To this end, sparqlPuSH relies on the aforementioned PubSubHubbub protocol to broadcast these updates.
In a nutshell the sparqlPuSH consists in the following steps:

1. to register the SPARQL queries related to the updates that must be monitored in a RDF store,
2. to broadcast changes when data mapped to these queries is updated in the store.

Moreover, it can be used as an interface on the top of any SPARQL endpoint and also comes with an ARC2[13] interface. Finally, this push approach can become a default model in various RDF store implementations, enabling more capabilities to monitor, in real-time, changes related to the RDF data.

3 Use Cases

We present the uses cases which are collected from the breakout session of the W3C LOD Track at WWW 2010[14]. Overall, we identified four general use cases which we describe in detail in the remainder. The use cases are ordered by increasing complexity to handle and process changes.

3.1 Use Cases UC 1: Synchronisation

A dataset consumer wants to mirror or replicate (parts of) a dataset from the LOD cloud. Ideally, the consumer wants to be informed about the statements that have been added/removed at a certain time point. The notification about the changes enables him an efficient way to keep his dataset up-to-date. Most commonly, the synchronisation is either for a single data source or for a set of data sources.

Real World Example. The semantic web index, Sindice, wants to keep its index always updated with the current version of the available LOD datasets, for instance DBPedia. A notification from the DBpedia publishers enables the index systems to decide at which time they want to update their index, instead of actively checking periodically if there exists a new version of a dataset dump. Ideally, Sindice would get a notification about a change of a dataset in general, and can request details about the changes to decide if it is necessary to perform the update immediately or at late point in time.

Requirements. The requirements to fulfill this use case are: The dataset publisher needs a dynamic description [DD] of the dataset which allows consumers to learn about high-level dynamics and the communication mechanism [CM] to learn about new changes. Further, the consumer has to be able to learn about this description by using a discovery mechanism [DM]. The publisher has to provide the change description [CD] which contains machine readable and understandable information about what and how much has changed. Finally, all the above requirements should be compliant with the Architecture of the Web [10] [CW] and have to scale to the size of the Web [SW].

3.2 Use Cases UC 2: Smart Caching

A developer uses one or more datasets from the LOD cloud in her application(s). Rather than implementing custom-code for keeping the local data in the application up-to-date (HTTP-level, dataset-level), the developer wants to use a *smart cache* that offers the functionality as required (e.g., 304/HTTP-level for small set of resources, notification for bulk-updates).

[13] `http://arc.semsol.org/`

[14] `http://www.w3.org/2010/04/w3c-track.html`

Real World Example. The execution of SPARQL queries directly over the LOD Web guarantees: on the one hand always fresh results, but is, on the other hand, very time and resource consuming. Each query triple pattern is executed directly over the dereferenced content of the URI constants in the triple patterns. The integration of a smart cache into the query processor could significantly increase the performance of such a system. The smart cache could be able to store statements or the whole content of resources which are rather very dynamic and frequently appear in queries. Such statements could be `rdf:type` statements.

Requirements. The requirements to fulfill this use case are the same as for UC1 with additionally: The dynamic and change descriptions of the datasets have to contain information for different **granularity levels [GL]**, e.g., information about changes at a statement, source or dataset level.

3.3 Use Cases UC 3: Link Maintenance

In many scenarios, we can integrate information from various datasets by using or creating links between the these datasets. A crucial point for any application which relies on these links is the problem that links can change or resources can disappear or moved. The application should be able to learn if a link type changed (e.g. the relationship between two instances was refined) or if the linked resources vanished or its identifier changed.

Real World Example. A music website enriches the information it provides about bands (for example `http://dbpedia.org/resource/Green_Day`) and artists with multilingual biography information retrieved from DBpedia. The website keeps the retrieved biographies in a local cache and updates them in regular intervals (e.g., once a day). Additionally it exposes its local information as Linked Data on the Web not including the cached biographies, but links to the corresponding DBpedia resources. The web application must be informed whenever the dependent resource (a representation of it) at DBpedia changes. Especially, if the resource is updated, deleted, or moved to another URI location (`http://dbpedia.org/resource/Green_Day_(band)`). If the web application is unaware of the remote changes it risks to (i) loose the ability to updated its cached biographies if remote resource become unavailable or (ii) expose dead or semantically invalid links in its local linked dataset. Additionally, it would be desirable to be informed about new artists and band biographies becoming available in the course of time.

Requirements. The requirements to fulfill this use case are the requirements from UC1 and UC2 - that is the explicit representation of the dynamics and changes (CD & DD) and the ability to discover the descriptions (DM) and which mechanism is used to communicate the changes (CM) in different granularity levels (GL). Further, for such a use cases the methods should be compliant to the architecture of the WWW and be scalable (CW & SW).

3.4 Use Cases UC 4: Vocabulary Evolution and Versioning

A given LOD dataset contains a set of resources that conform to a particular vocabulary. In other words, the vocabulary provides classes and properties for expressing the data of the dataset. Whenever the vocabulary changes (evolves), i.e. a new version of the vocabulary is available, there has to be some support for the propagation of the vocabulary/ontology changes to the dataset. The resources of the dataset have to be updated, and in this way they will be conformed to the new version of the vocabulary.

Real World Example. The *FOAF ontology*[15] provides classes and properties for expressing some DBpedia resources. Last January, a new version of *foaf ontology* was released. This new version updates some properties, e.g., `foaf:givenname` to `foaf:givenName`. A notification of the new version of the ontology, with its changes, is sent to the related datasets, including DBpedia. Each dataset will decide when perform the propagation of those changes.

Requirements. In this use case the requirements to fulfill are the same of UC1, but taking into account the ontology/vocabulary as well: (DD) the explicit representation of the ontology/vocabulary changes, (DM & CM) the notification and change propagation approaches that allow to discover and register/subscribe the changes of the vocabulary/ontology, and (GL) the selection of the right granularity level of the changes.

4 Dataset Dynamics Requirements

We identified already the core requirements to solve our presented use cases, clustered into: *description*, *mechanism* to discover and communicate changes for certain *granularity levels*. Further, dealing with Linked Data according to the four principles puts also some *architectural* requirements in place.

Description [DD & CD]
Dealing with dataset dynamics, we clearly need a way to describe that a dataset is dynamic and how a data consumer can learn about that something has changed and, in addition, what has changed. The descriptions should be machine readable and even more important understandable - that is that the description should use the same set of RDF vocabularies. Further, the descriptions should contain the important attributes: 1) general information about the expected frequency of changes 2) information and pointers to notification mechanisms. In addition, the following optional and welcomed attributes are: 3) information about the average change volume and 4) the type of change,e.g., most of changes are updates of available information or the add of information.

Granularity Levels [GL]
The majority of our use cases need or could benefit from different granularity levels by how changes are detected and communicated. Certain applications require change notifications on the statement level (e.g. the link type or the object value changed), whereas for others it is sufficient enough to know that there appeared a change in a dataset. For coarse grained levels it might be necessary to also know about the change fraction; e.g. 50% of the dataset changed. In more detail, we identified the necessity for the following change levels:

Mandatory Levels

d Dataset level - the dataset *DS* changed
r Resource level - the resource *URI* changed
s Statement level - the statement (*URI URI VALUE*) was deleted or added (this covers also update operations)

Optional Levels

g Graph Structural Level - e.g. adding removing named graphs which can be collections of statements from different sources

Communication Mechanism [CM]
We need scalable mechanisms to communicate the change event and the changes itself. Data consumers should be able to either actively check (pull) or listen (push) for such events and the change descriptions. A typical publisher/subscriber mechanism with

[15] `http://xmlns.com/foaf/spec/`

different communication channels would be ideal. There should be a communication channel that contains information that something has changed and how much has changed. Another grouped of channels should then contain detailed information about the changes for different granularity levels.

Discovery Mechanism [DM]
Dataset consumers needs mechanism to discover and learn about the change description of a dataset and which mechanism exists to learn about changes and what has changed. Ideally, mechanism like the Linked-based Resource Description Discover Protocol (LRDD) is used. LRDD is a collection of three link methods and a common relation type for associating a descriptor to the resource it describes.

Web Architecture and Scalability [SW,CW]
Approaches which deal with dynamics of Linked Data Webs (either the open Web or in intranets) should use techniques that are compliant to the architecture of the WWW [10] and should scale to the size of the Web. In addition, the approaches should ideally be distributed, provide a publisher/subscriber model where possible and allow batch operations.

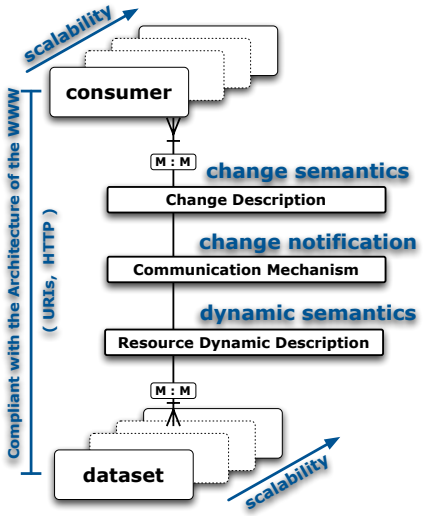

Fig. 1: Abstract Dataset Dynamics Stack.

Abstract Dataset Dynamics Stack
Next, we introduce the abstract and high-level dataset dynamics architecture and technology stack as depict in Figure 1. We have *datasets* which are undergoing changes as we already motivated above.
This information is consumed by software agents and humans, representative examples of such producer/consumer interactions are motivated in our use cases. The architecture stack exists of two description layers–both on the dataset and consumer side– and communication mechanisms which enables the interaction between consumers and dataset providers.
A dataset needs to describe its general dynamic attributes (e.g., the average expected change frequency) and how a consumer can learn about occurring changes (cf. dynamic semantics in Figure 1). The consumer, on the other hand, needs to learn and understand that a dataset has undergoing a change and what exactly has changed (cf. change semantics in Figure 1).

Further, we can see that we have to deal with many-to-many relationship between consumers and datasets which requires that the communication mechanisms needs to be highly scalable on both sides. We can expect to have potentially millions of consumers and thousands of datasets, millions of resources and billions of triples. So each dataset can be consumed by millions of agents and each agent can consume thousands if not millions of information pieces.
In addition, the communication mechanisms should be compliant to the architecture of the WWW which means concrete solutions have to make use of URIs and the HTTP protocol layer.
We deliberately omitted here the discovery layer which allows consumers to automatically locate the resource dynamic description of a dataset. The aspects of resource description discovery is surveyed in [25].

5 Survey of Approaches

In this section we describe the most significant approaches that treats the aspects related to datasets dynamics, taking into account the identified requirements in Section 4. We have grouped the approaches into the two identified layers of Figure 1, descriptions and communication mechanisms.

5.1 Descriptions

There are several approaches for describing the dataset dynamics and the semantics of changes in a dataset. Next, we present the most representative approaches we have found in the literature. The summary of this survey and the comparison with the requirements are listed in Table 1.

Descriptions	**Dynamic description**	**Change description**	**Granularity levels**
DaDy	√	-	-
DSNotify Eventset Vocabulary	√	√	**s**
Talis Changeset Vocabulary	√	√	**r**
OWL 2 change ontology	√	√	**d,r**
CHAO	√	√	**r**

Table 1: Summary of description proposals and their coverage of the requirements. Granularity levels: dataset (**d**), resource (**r**) and statement (**s**) level.

Dady
The Dataset Dynamics Vocabulary, DaDy [3], can represent information about the regularity (regular, irregular) and frequency (no, low, mid, high) of updates and provide a link to the update notification source URI. It is designed to be used with voiD [2].

DSNotify Eventset
The DSNotify Eventset Vocabulary [4] [12], is a vocabulary for change events in linked data sources, and can be used to describe timely-ordered sets of events that modify resources in linked data resources. Eventsets[16] are associated with two voiD:Datasets a source and a target dataset.

Talis Changeset
The Talis Changeset Vocabulary [5] defines a set of terms for describing changes to resource descriptions. In the context of this vocabulary, a resource description is the set of triples that in some way include a description of a resource. Moreover, the vocabulary introduces the notion of a *ChangeSet* which encapsulates the delta

[16] A Eventset is a container of events that occur in a dataset.

between two versions of a resource description. The delta is represented by two set of triples: additions and removals. A ChangeSet can be used to modify a resource description by first removing all triples from the description that are in the removals set and adding the triples in the additions set.

OWL 2 change ontology
The OWL 2 change ontology [1] is a fined-grained taxonomy of ontology changes that considers the lowest-level atomic operations that can be performed in an ontology, but also on other abstraction levels, for example: atomic, entity and composite changes. This ontology allows to describe on a fine grained level how an ontology has changed from one version to another.

CHAO
The Change and Annotations Ontology, CHAO [13], represents ontology changes within the ontology-evolution system of Protégé. The ontology consists of two parts. The basis is an ontology of basic change operations and there is an extension that defines complex change operations. Instances of these ontologies record information about chances including meta information about them, e.g., author, timestamp, annotations, etc.

5.2 Communication Mechanisms

The set of communication mechanisms include discovery mechanism, communication protocols and delta computations. As mentioned earlier, we exclude the discovery and delta computation mechanisms from this survey since they deserve a comprehensive survey on its own. The summary of this survey and the comparison with the requirements are listed in Table 2.

Mechanisms	**Discovery mechanism**	**Granularity level**	
		Pull	**Push**
Atom	√	**d,r**	**-**
PubSubHubbub	√	**d,r,s**	**d,r,s**
SDShare	√	**d,r,s**	**-**
OAI-PMH	√	**d,r**	**-**
PingTheSemanticWeb	√	**-**	**r**
SemanticPingback	**-**	**-**	**r**
Web Hooks	√	**-**	**d,r**
RDFSync	-	**d,r,s**	**-**
SPAURL	√	**-**	**s**
DSNotify	√	**d,r,s**	**-**

Table 2: Summary of communication mechanism proposals and their coverage of the requirements. Granularity levels: dataset (**d**), resource (**r**) and statement (**s**) level.

Current approaches for the communication between dataset consumers and publisher can be split into two kinds of notification mechanisms: 1) a *pull* mechanism (e.g., feed subscription, web crawlers or monitored queries) and 2) a *push* mechanism which are mainly implemented by a publisher/subscriber model. A detailed comparison about push vs. pull mechanisms is given by Bhide et.al [7]. Further, the authors propose a combination of a push and pull based approach as an nearly optimal solution for the communication process (cf. PubSubHubbub). A hybrid push and pull approach seems to be able to establish a stable system which can deal with arbitrarily large numbers of subscribers and changes and rapid changes of the dataset.

Pull Based Approaches

In short, Pull based approaches have to deal with a large communication overhead for the messages exchanged and a large number of clients and further, have problems to deal with rapidly changing data.

Atom

Atom [17] is an XML-based Web content and metadata syndication format, and an application-level protocol for publishing and editing Web resources belonging to periodically updated websites. Atom is a relatively recent spec and is much more robust and feature-rich than RSS [17].

SDShare

The protocol for the Syndication of Semantic Descriptions, SDShare [20], defines how a RESTful web service can publish a series of web accessible feeds that describe snapshots and changes to collections of semantic descriptions.

OAI-PMH

The Open Archives Initiative Protocol for Metadata Harvesting, OAI-PMH [14], provides an application-independent interoperability framework based on *metadata harvesting*. There are two classes of participants in the OAI-PMH framework: 1) Data Providers administer systems that support the OAI-PMH as a means of exposing metadata; and 2) Service Providers use metadata harvested via the OAI-PMH as a basis for building value-added services.

RDFSync

RDFsync [11] is an approach for the efficient synchronization of RDF models. Because of the RDF semantics, RDF models cannot be efficiently synchonized by the rsync[17] or similar algorithms. RDFSync is based on the decomposition of a model into Minimum Self-Contained graphs (MSGs).

Push Based Approaches

Basically, push based mechanisms can deal very efficiently with high frequently changing data, but on the other side these push mechanisms have to maintain the list subscriber and the states of open connections which can cause also scalability problems. Another problem is that the notification messages can be potentially very large (especially if a client requests changes on a statement level).

PingTheSemanticWeb

PingTheSemanticWeb [15] is a web service archiving the location of recently created/updated RDF documents on the Web. If one of those documents is created or updated, its author can notify PTSW that the document has been created or updated by pinging the service with the URL of the document.

Semantic Pingback

Semantic Pinback [23] tackles the quality, timeliness and coherence as well as direct end user benefits of the emerging Linked Data Web. Semantic Pingback extends the well-known Pingback method, which is technological cornerstone of the blogosphere. It is based on the advertising of an RPC service for propagating typed RDF links between Data Web resources.

WebHooks

WebHooks [27] are HTTP callbacks which uses HTTP POST operations to learn that something happens and for notifications. Clients just register a webhook to a URL and receive notifications whenever a event occurs. The publisher just needs to send a HTTP POST to the specific URL with the event description.

[17] rsync is a software application for Unix systems which synchronizes files and directories from one location to another while minimizing data transfer using delta encoding when appropriate.

SPARUL
The updated language for RDF graphs, Sparql/Updates [26], is able to express updates to an RDF store. It is intended to be an standard mechanism by which updates to a remote RDF store can be described, communicated and stored.

5.3 Comparison Overview

After having analyzed the existing descriptions and communication mechanisms, that treats the aspects related to datasets dynamics, we present the comparison of the existing research works according to the identified requirements described in Section 4.
Regarding the existing descriptions, we can state that: **DD** is covered by all the approaches; **CD** is covered by the Talis ChangeSet, the OWL 2 change ontology, and the Change and Annotations Ontology; **GL** is covered by (1) DSNotify Eventset Vocabulary at statement level, (2) Talis ChangeSet at the resource description level, (3) OWL 2 change ontology at entity and composite changes, and (4) Change and Annotations Ontology, at basic and complex operations. Regarding the communication mechanisms, we can say that most fall into either push-based or pull-based approaches, with only few supporting both. However, the communication mechanisms are not mature enough.

6 Conclusion

The importance of this work is motivated by the problematic of web-scale handling dataset dynamics; the discovery of the description of the dynamic of a dataset, how a data consumer can learn about changes and the actual protocol to communicate changes and methods to efficiently compute changes between two versions of a dataset. This survey encompasses that there exists not a clear and solid solution to solve the highlighted problems of discovering, describing and communicating the change dynamics of Linked Data (re)sources.

We compared a number of deployed systems and technologies concerning requirements we derived from real world use cases the community came up with. Further, we introduced an abstract dataset dynamics stack, which provides the community with a framework and can potentially serve as a basis for further work into benchmarking and comparing solutions on a wider range.

Acknowledgements
The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n 256975 , LOD Around-The-Clock (LATC) Support Action.

References

1. Owl 2 change ontology, 2009. `http://omv.ontoware.org/OWLChanges`.
2. Vocabulary of interlinked datasets, 2009. `http://vocab.deri.ie/void/guide`.
3. Dataset dynamics (dady) vocabulary, 2010. `http://purl.org/NET/dady`.
4. Dsnotify eventsets: A vocabulary for change events in linked data sources, 2010. `http://dsnotify.org/vocab/eventset/0.1/`.
5. Talis changeset vocabulary, 2010. `http://vocab.org/changeset/schema`.
6. J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and implementation of schema evolution in object-oriented databases. In *SIGMOD '87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pages 311–322, New York, NY, USA, 1987. ACM.
7. M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy. Adaptive push-pull: Disseminating dynamic web data. *IEEE Transactions on Computers*, 51:652–668, 2002.

8. C. Bizer, T. Heath, and T. Berners-Lee. Linked Data – The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
9. S. Hellmann, C. Stadler, J. Lehmann, and S. Auer. Dbpedia live extraction. In *Proc. of 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, volume 5871 of *Lecture Notes in Computer Science*, pages 1209–1223, 2009.
10. I. Jacobs and N. Walsh. Architecture of the world wide web, volume one. World Wide Web Consortium, Recommendation REC-webarch-20041215, December 2004.
11. C. Morbidoni, G. Tummarello, O. Erling, and R. Bachmann-Gmr. Rdfsync: efficient remote synchronization of rdf models. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, G. Schreiber, and P. Cudr-Mauroux, editors, *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 533–546, Berlin, Heidelberg, November 2007. Springer Verlag.
12. P. Niko and H. Bernhard. Dsnotify: Handling broken links in the web of data. In *Nineteenth International WWW Conference (WWW2010)*, Raleigh, NC, USA, 2 2010. ACM.
13. N. F. Noy, A. Chugh, W. Liu, and M. A. Musen. Musen m.: A framework for ontology evolution in collaborative environments. In *In: 5th International Semantic Web Conference*, pages 544–558. Springer-LNCS, 2006.
14. OAI-PMH. The open archives initiative protocol for metadata harvesting, 2010. `http://www.openarchives.org/OAI/openarchivesprotocol.html`.
15. PingtheSemanticWeb. The open archives initiative protocol for metadata harvesting, 2010. `http://pingthesemanticweb.com/`.
16. pubsubhubbub. A simple, open, web-hook-based pubsub protocol, 2010. `http://code.google.com/p/pubsubhubbub/`.
17. rfc4287. The atom syndication format, 2010. `http://www.ietf.org/rfc/rfc4287.txt`.
18. J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37:383–393, 1995.
19. RSS. Really simple syndication, 1999. `http://www.rss.com/`.
20. SDShare. Protocol for the syndication of semantic descriptions, 2010. `http://www.egovpt.org/fg/CWA_Part_1b`.
21. Sitemap. Sitemap protocol, 2008. `http://sitemaps.org/`.
22. SparqlPuSh. pubsubhubbub (push) interface for sparql endpoints, 2010. `http://code.google.com/p/sparqlpush/`.
23. S. Tramp, P. Frischmuth, T. Ermilov, and S. Auer. Weaving a Social Data Web with Semantic Pingback. In *Proceedings of the EKAW 2010 - Knowledge Engineering and Knowledge Management by the Masses; October, 2010*, pages 135–149, 2010.
24. G. Tummarello, E. Oren, and R. Delbru. Sindice.com: Weaving the Open Linked Data. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, pages 547–560.
25. J. Umbrich, M. Hausenblas, P. Archer, E. Hammer-Lahav, and E. Wilde. Discovering resources on the web - a comparison of discovery mechanism for the web of data and the web of documents. Technical Report 1, Linked Data Research Centre, 8 2009. see also http://uldis.deri.ie.
26. S. Update. A language for updating rdf graphs. w3c member submission, 2010. `http://www.w3.org/Submission/SPARQL-Update/`.
27. Webhooks. Webhooks, 2010. `http://www.webhooks.org/`.
28. R. Zicari. A framework for schema updates in an object-oriented database system. In *Building an object-oriented database system: the story of 02*, pages 146–182. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.